Remarks/Arguments

The examiner has requested that the references related to the application cited in the specification and filed on 9/5/00, paper number 4, be updated to show the relevant status. Accordingly, the Applicant has amended the status of the related applications where appropriate. No new matter is added by these amendments.

Claims Rejections under 35 U.S.C. 103(a)

The Examiner had previously rejected claims 11-33 under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,360,230 to Chan et al. (hereinafter referred to as "Chan"). In the Applicant's amendment mailed December 11, 2003, filed concurrently with a Request for Continued Examination, the Applicant presented a variety of arguments distinguishing the pending claims from the Chan reference. In his office action of February 25, 2004, the Examiner continued to reject the pending claims, but made no further mention of the Chan reference. Accordingly, the applicant has assumed that the Examiner found the Applicant's prior arguments persuasive, and thus removed his rejection of Claims 11-33 over Chan. If the Applicant is mistaken in this assumption, the Applicant respectfully requests that the Examiner bring such to the Applicant's attention.

In the Applicant's amendment mailed December 11, 2003, the Applicant also presented numerous amendments to the Claims. The Applicant states for the record these amendments, together with any and all prior amendments to the Claims made by the Applicant, have been made for the sole purposes of presenting the Claims in a format acceptable the United States Patent and Trademark Office and to provide greater clarity, and have not been made for the purpose of distinguishing the Claims from the prior art. **[Ed: based on my reading of the prior arguments and amendments, I believe this is accurate. If it is, I think it is a useful statement to make, given Festo. However, I am not sure I have seen every office action and response, so you may want to double check with the attorney(s) who wrote the previous office actions to make sure]**

Claims 11-33 remain pending, with Claims 11 and 31 being independent claims.

The Examiner rejects claims 11-16, 18-19, and 25-33 as being unpatentable over Havens et al., US Patent No. 5,732,263 (hereinafter referred to as Havens) in view of Halviatti et al., US Patent No. 5,475,843 (hereinafter referred to as Halviatti). The applicant notes that throughout the Examiner's rejections, the Examiner has repeatedly misstated the claims as set forth in the Applicant's December 11, 2003 amendment. These misstatements appear to be a result of portions of the amended claims being read as if subject matter deleted by the amendment, indicated by a ~~strikethrough~~, was not in fact deleted. Accordingly, the Applicant has assumed that the Examiner examined the claims in their amended form, and that these misstatements were the result of word processing errors, and were unintentional. If the Applicant is mistaken in this assumption, the Examiner is respectfully requested to bring this to the attention of the Applicant.

The Examiner recites that Havens teaches "a method for dynamically generating an object class in a computer system comprising the steps of: creating a global generic class" (citing column 8, lines 15-50 and fig. 10) "having a first member being related to at least one attribute and a second member being related to at least one method" (citing column 8, lines 28-67), and "instantiating the global generic class in order to have generate the object class" (sic) (citing column 9, lines 1-50). The Examiner concedes that "Havens does not explicitly teach the step of wherein at least one member is an instance of a generic class, the generic class having at least a name as an attribute," but asserts that Halviatti teaches this step, citing column 6, lines 60-65 and column 30, lines 15-50. The Examiner further asserts that it would have been obvious to combine the teaching of Havens and Halviatti's system because "Halviatti's member in an instance of a generic class would provide more choices of attributes names." (sic) The Examiner has thus mischaracterized the teachings of both Havens and Halviatti, and has failed to set forth a prima facie case of obviousness.

At the outset, the Examiner's assertion that Havens teaches "a method for dynamically generating an object class" is not accurate. While Havens does use the term "object class generation" in Figure 10 and at column 8, lines 38-39, and intimates that the base code can "create" object classes at column 7, lines 17-20, it is clear from the remainder of the specification that Havens does not mean that the software itself "generates" or

"creates" the object class. Rather, Havens' software creates selector IDs and class IDs that allow new object classes created by a user to interact with a base program. Thus, while Havens' software may associate new object classes with a base program, as described by Havens, the method for forming the new object classes is not dynamic; it is static.

This is not surprising, as the problem Havens is trying to solve is not directed toward the problem of forming object classes, dynamically or otherwise. Rather, as stated by Havens, the purpose of Havens' disclosure is to "allow users to customize object oriented programs whereby new object classes can be created and utilized in conjunction with the base program without requiring modification to the base program." See column 2, lines 12-17. Thus, Havens' concern is simply that new object classes interact successfully with the base program, and is not directed at methods and techniques for creating those object classes.

In contradistinction, the present invention both teaches and claims a method through which new object classes are created dynamically. While the present invention thus accomplishes the objective of the Havens' disclosure, (new object classes dynamically created using the present invention do, in fact, interact successfully with the base program), the converse is not true. Havens' object classes may interact successfully with the base program, but Havens does not produce these new object classes dynamically.

At best, Havens produces new object classes by first cloning an existing object class, and then statically modifying it. In the paragraph bridging columns 7 and 8, Haven makes this point explicitly. Havens states:

> "Referring now to FIG. 7, the addition of a new object class after build-time, according to the present invention, is illustrated. A new subclass "VICE PRESIDENT" has been added as a subclass of the subclass "EXECUTIVE." "VICE PRESIDENT" is assigned a selector ID and a class ID, and the class ID is included within a subclass table therein. Although "VICE PRESIDENT" initially inherited the attributes and methods of "EXECUTIVE", as shown in FIG. 7, "VICE PRESIDENT" now includes only one attribute and *no methods* from "EXECUTIVE." *The method*

"COMPUTE NEW BONUS" encapsulating the attributes of "VICE PRESIDENT" is a method not found anywhere in the hierarchy. (italics added)

How then does Havens create methods for the new object class "VICE PRESIDENT"? Havens doesn't. Havens depends on the *user* to create the new object class. Havens explains this in the comments to the pseudo-code at columns 23 and 24, lines 20-25, where Havens' states:

> The client code is presented to show what a user would have to provide to sub-class one (of) our provided classes. The code assumes that the client programmer has determined that they wish to subclass the IBM-provided fWorkStation1 class]. They have decided to override ALL of the behavior of that class – so they use the selector for the fWorkStation1 class which is WORKSTATION1.

It is thus the *user*, not the *software*, that creates the new object classes. Havens reiterates the point in the paragraph bridging columns 9 and 10:

> "Assume that a user is not satisfied with the method of computing bonuses, "COMPUTE BONUS," included within the vendor-provided object class "EXECUTIVE." The user decides to generate a new object class entitled "VICE PRESIDENT" comprising *a completely different method of calculating the yearly bonus*. By registering "VICE PRESIDENT" in the user class portion of the registration table with the same selector ID as "EXECUTIVE," any base program code *or user code* that attempts to instantiate an "EXECUTIVE" object, will instantiate a "VICE PRESIDENT" object. Any requests to perform "COMPUTE BONUS" on an object will automatically *call the new method* "COMPUTE NEW BONUS." The base program will recognize only the first selector ID having the value transmitted in an instantiation request (block 302). All other object classes registered with the registration table with the same selector ID will be ignored. As a result, *user-created object classes* obtain control prior to any vendor-provided object classes having the same selector ID." (italics added)

Havens thus makes it clear that new object classes such as "VICE PRESIDENT" must have methods written entirely by the user, since the new method "is a method not found anywhere in the hierarchy." Further, Havens makes clear that the object classes are "user-created." This is the very definition of "statically" creating object classes. Havens is thus entirely unconcerned with the objective of the present invention, dynamically creating object classes, and does nothing to accomplish that objective.

Since Havens is not concerned with "dynamically generating an object class," it is not surprising that Havens explicitly teaches directly away from the requirement of claim 11, (and dependant claims 12-16, 18-19, and 25-30), that the global generic class has "a second member being related to at least one method." In statically building a new object class, Havens has no need for such interoperability. Havens thus separates it completely from any existing object class, teaching that the new method "is a method not found anywhere in the hierarchy." This is an explicit contradiction of the requirement of the present invention that the global generic class has "a second member being related to at least one method." As noted by the Federal Circuit, "references that teach away cannot serve to create a prima facie case of obviousness." In re Gurley, 27 F.3d 551, 553, 31 USPQ2d 1131, 1132 (Fed. Cir. 1994). Accordingly, contrary to the Examiner's assertion, Havens does not provide a teaching or suggestion for this feature of the claims, and cannot, therefore, form the basis for a prima facie case of obviousness under 35 U.S.C. 103(b). "To support the conclusion that the claimed invention is directed to obvious subject matter, either the references must expressly or impliedly suggest the claimed invention or the examiner must present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references." Ex parte Clapp, 227 USPQ 972, 973 (Bd. Pat. App. & Inter. 1985). The Havens reference cannot possibly be interpreted as "expressly or impliedly suggesting" even this single step, much less the claimed invention, as Havens teaches the exact opposite; that the new method "is a method not found anywhere in the hierarchy."

The Examiner's assertion that Havens teaches "instantiating the global generic class in order to have generate the object class" (sic) is also entirely without merit. The Examiner cites column 9, lines 1-50 as providing support for this teaching. However, Havens software never "generates" an object class, and in fact makes this precise point abundantly clear at column 9, lines 1-50. In reference to object classes created by the user (which are created statically, as shown above) Havens states:

> "According to the present invention, the base program is able to instantiate objects from classes it is not aware of via the registration table which retains the selector IDs for every subclass derived from the exemplar class. All vendor-provided classes, existing at build-time are registered in the vendor class portion of the table. All classes generated after build-time are registered in the user class portion of the table."

Thus, to instantiate classes generated after build time, according to Havens, the object classes must already be in existence. Otherwise, they could not be stored in the user class portion of the table. Havens is therefore not "generating" anything. He is merely showing how the base code is able to communicate with new object classes which were written by a user after compiling the base code, and which were not, therefore, originally a part of the base code. These object classes are not "dynamically generated" as a result of instantiating a "global class," as in the present invention. The user created object classes in Havens are already in existence, and are "user created," not "dynamically generated." The only point Havens is making in this section of the specification is that the base code is able to communicate with these "user created" object classes, even if the base code does not know that these user created object classes exist. But it is crystal clear that they must already exist. As stated by Havens:

> "When a matching selector ID is found, an object is instantiated from the object class having the same name as that linked to the found selector ID (block 306). Consequently, the exemplary class need not know that the requested object class exists. As long as an object class name and a corresponding selector ID is registered

with the registration table, the base program can properly handle the instantiation of objects from unknown object classes."

Accordingly, Havens does not remotely disclose the step of "instantiating the global generic class to generate the object class," as required by claims 11, and claims 12-30 by virtue of dependency. Havens cannot, therefore, form the basis for a prima facie case of obviousness under 35 U.S.C. 103(b) as Havens provides no basis whatsoever for the suggestion or teaching of this step.

The Examiner has also mischaracterized the teaching of Halviatti. Halviatti is entirely unconcerned with the creation of object classes whether statically, dynamically, or otherwise. Instead, Halviatti is directed at a method for testing the reliability of software programs, particularly computer-aided software testing systems which "assist a Quality-Assurance engineer and software developers with the testing of Graphical User Interface (GUI) software programs operative on digital computers." The Examiner asserts that Halviatti "teaches the step of wherein at least one member is an instance of a generic class, the generic class having at least a name as an attribute."

The Examiner's citation to column 6, lines 60-65 as supporting this assertion appears to be an error. Column 6, lines 60-65 is a discussion of the drawbacks of sequential testing programs of the prior art. No mention whatsoever is made concerning any class, attribute, or member in any context whatsoever. The Examiner's further citation to column 30, lines 15-50 seems likewise misplaced. In this section Halviatti does describe attributes for a database, stating "The Resource Database is where expected attributes of these GEM instances, such as label text, size, color, and the like, are stored." Notably, contrary to the Examiner's assertion, "name" is not one of the attributes listed by Halviatti, nor is the "Resource Database" a generic class. Even if they were, however, it is important to note that they would still be in a context entirely distinct from the present invention. As with Havens, Halviatti's software is not "generating" object classes. Even if Halviatti taught a "generic class having at least a name as an attribute" such a teaching would still be entirely irrelevant to the present invention as claimed, as it would in no way be connected to the creation of a

"global generic class" or the instantiating of that "global generic class" to "generate an object class." As such, Halviatti cannot provide the teaching that "at least one member is an instance of a generic class, the generic class having at least a name as an attribute" in the context of dynamically generating an object class.

To establish a prima facie case of obviousness based on a combination of the content of various references, there must be some teaching, suggestion or motivation in the prior art to make the specific combination that was made by the applicant. In re Raynes, 7 F.3d 1037, 1039, 28 USPQ2d 1630, 1631 (Fed. Cir. 1993); In re Oetiker, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992). Obviousness can not be established by hindsight combination to produce the claimed invention. In re Gorman, 933 F.2d 982, 986, 18 USPQ2d 1885, 1888 (Fed. Cir. 1991). As discussed in Interconnect Planning Corp. v. Feil, 774 F.2d 1132, 1143, 227 USPQ 543, 551 (Fed. Cir. 1985), "it is the prior art itself, and not the applicant's achievement, that must establish the obviousness of the combination." Accordingly, since neither Havens nor Halviatti teach or suggest the step of "instantiating the global generic class to generate the object class," show the use of "a second member being related to at least one method," or provide a teaching that "at least one member is an instance of a generic class, the generic class having at least a name as an attribute," a prima facie case of obviousness cannot possibly be constructed, as there cannot possibly be a suggestion to combine these various elements of the claim since they are entirely absent from the prior art. The Applicant therefore respectfully requests that the Examiner remove his rejection of Claim 11.

With respect to Claim 12, the Examiner asserts that Havens teaches the step of "wherein the first member is an attribute of the global generic class, said first member being an instance of a generic attribute class." Havens does not, in fact, show such a step, as Havens merely shows the step of "assigning and registering a selector ID, and assigning and registering a class ID." See column 8, lines 38-42. But even if Havens did provide that "the first member is an attribute of the global generic class, said first member being an instance of a generic attribute class," as described by Havens, the object class is not dynamically "generated" by the operation of the software, it is instead "derived" from an existing

subclass, and then statically created by a user. Accordingly, the Applicant respectfully requests that the Examiner remove his rejection of Claim 12.

With respect to Claims 13 and 14, the Examiner asserts that Havens teaches the step of "wherein the second member is a method of the global generic class, said second member being an instance of a generic attribute class." Havens does not, in fact, show such a step, as Havens explicitly insures that any method in a new object class "is a method not found anywhere in the hierarchy." Even if Havens did show such a step, Havens still does not show the step of "instantiating the global generic class to generate the object class" as Havens creates the methods of new object classes statically. Accordingly, the Applicant respectfully requests that the Examiner remove his rejection of Claims 13 and 14.

With respect to Claims 15 and 16, the Examiner asserts that Havens teaches the step of "wherein the method of the global generic class is defined by at least one parameter derived from an instance of a generic parameter class." Assuming, *arguendo*, that Havens' step of "assigning and registering a selector ID, and assigning and registering a class ID" could be interpreted as "a parameter derived from an instance of a generic parameter class," the object class of Havens is still not dynamically "generated" by the operation of the software; it is instead "derived" from an existing subclass, and then statically created by a user. Accordingly, the Applicant respectfully requests that the Examiner remove his rejection of Claims 15 and 16.

With respect to Claims 18 and 25-30, the Examiner asserts that Havens teaches the step of "wherein the method is implemented in a command interface of the computer system." As described above, Havens does not show the method implemented at all, as Havens does not show the step of "instantiating the global generic class to generate the object class," and Havens does not show the use of "a second member being related to at least one method," so even if Havens does use a command interface, it is still insufficient to form the basis of a prima facie case of obviousness under 35 U.S.C. 103(a). Accordingly, the Applicant respectfully requests that the Examiner remove his rejection of Claims 18 and 25-30.

With respect to Claim 19, the Examiner asserts that Havens teaches the step of "wherein the method is implemented the global generic class and the generic class is created by a designer who is a computer expert and using the command interface used for the computer system by a user who may not be a computer expert uses the command interface to instantiate the global generic class created by the designer to generate the object class." (sic) As described above, Havens does not show the method implemented at all, as Havens does not show the step of "instantiating the global generic class to generate the object class," whether by a user, or by a computer expert, or by the software itself. In fact, at the precise location cited by the Examiner as support for this proposition, column 7, lines 12-24, Havens makes it plain that "when object classes are modified or replaced, the base program needs some method of recognizing and validating instances created from these object classes. The base program will not be aware of objects created from object classes modified or added after build-time." As taught by Havens, the base program not only does not generate the object classes, it "will not be aware of objects created from object classes modified or added after build-time." Thus, Havens is insufficient to form the basis of a prima facie case of obviousness for this step under 35 U.S.C. 103(a). Accordingly, the Applicant respectfully requests that the Examiner remove his rejection of Claim 19.

The Examiner rejects claims 31-33 with the recitation that they are "the system of claim 11." Accordingly, the Applicant hereby reiterates the explanation given above, that neither Havens nor Halviatti teach or suggest the step of "instantiating the global generic class to generate the object class," show the use of "a second member being related to at least one method," or provide a teaching that "at least one member is an instance of a generic class, the generic class having at least a name as an attribute" in the context of dynamically generating an object class. Accordingly, the combination of Havens and Halviatti is insufficient to form the basis of a prima facie case of obviousness for any of these steps under 35 U.S.C. 103(a), and the Applicant respectfully requests that the Examiner remove his rejection of Claims 31-33.

The Examiner rejects claims 17 and 20-24 under 35 U.S.C. 103(a) as being unpatentable over Havens in view of Halviatti, and further in view of Stutz, US Patent No. 5,485,617 (hereinafter Stutz). The Examiner concedes that Havens and Halviatti do not teach "the step of automatically generating the global generic class and the generic class by means of a tool having respective dialogue boxes defining attributes of these classes, including the name attribute of the generic class." The Examiner asserts, however, that Stutz teaches this step. In making this assertion, the Examiner misstates the teaching of Stutz in the same manner that the Examiner misstated the teaching of Havens. As with Havens, Stutz does not dynamically generate an object class. Just as with Havens, Stutz provides a method for "dynamically generating object connections" rather than the object classes themselves. As such, Stutz does not teach or suggest the step of "instantiating the global generic class to generate the object class," show the use of "a second member being related to at least one method," or provide a teaching that "at least one member is an instance of a generic class, the generic class having at least a name as an attribute" in the context of dynamically generating an object class. Accordingly, even if Stutz provides a teaching of a "dialogue box," it does not teach or suggest the combination of the dialogue box with any of the aforementioned steps. Thus, since none of Havens, Halviatti and Stutz teach any of these steps, the combination of Havens, Halviatti and Stutz is insufficient to form the basis of a prima facie case of obviousness under 35 U.S.C. 103(a). The Applicant respectfully requests that the Examiner remove his rejection of Claims 17 and 20-24.

Summary

Applicants respectfully request favorable reconsideration of this application. Claims 11-33 are pending, with Claims 11 and 31 being the only independent claims. Applicants have made an earnest attempt to place the above referenced application in condition for allowance and action toward that end is respectfully requested. Should the Examiner have any further observations or comments, he is invited to contact the undersigned for resolution.

To the extent necessary, Applicants petition for an extension of time under 37 CFR § 1.136. Please charge any shortage in fees due in connection with this application, including

extension of time fees, to Deposit Account No. 50-1165 (Attorney Docket No. T2147-906524) and credit any excess fees to the same Deposit Account.

Respectfully submitted,

MILES & STOCKBRIDGE P.C.

Date: May 4, 2004

By: _____
Edward J. Kondracki
Registration No. 20,604
Agent for Applicants

MILES & STOCKBRIDGE P.C.
1751 Pinnacle Drive, Suite 500
McLean, VA 22102-3833
(703) 903-9000